Adapting Event Embedding for Implicit Discourse Relation Recognition

Maria Leonor Pacheco, I-Ta Lee, Xiao Zhang, Abdullah Khan Zehady
Pranjal Daga, Di Jin, Ayush Parolia, Dan Goldwasser
Department of Computer Science, Purdue University
West Lafayette, IN 47907
{pachecog,lee2226,zhang923,azehady
daga,jind,aparolia,dgoldwas}@purdue.edu

Abstract

Predicting the sense of a discourse relation is particularly challenging when connective markers are missing. To address this challenge, we propose a simple deep neural network approach that replaces manual feature extraction by introducing event vectors as an alternative representation, which can be pre-trained using a very large corpus, without explicit annotation. We model discourse arguments as a combination of word and event vectors. Event information is aggregated with word vectors and a Multi-Layer Neural Network is used to classify discourse senses. This work was submitted as part of the CoNLL 2016 shared task on Discourse Parsing. We obtain competitive results, reaching an accuracy of 38%, 34% and 34% for the development, test and blind test datasets, competitive with the best performing system on CoNLL 2015.

1 Introduction

The CoNLL 2016 shared task focuses on Discourse Parsing. Building on the CoNLL 2015 task, this year teams were able to focus on a supplementary task, limited to sense classification of discourse relations, given their (gold) arguments (Xue et al., 2016). Identifying the sense is particularly challenging in the case of implicit relations, where explicit connective words (e.g., *however, but, because*) are not present. Last year, most submitted systems used algorithms traditionally applied for this task, such as Support Vector Machine (SVM) and Maximum Entropy classifiers learned over binary features as input representation. This included the best performing system, which reached an accuracy of 34.45 in the test data and an accuracy of 36.29 in the blind test data for implicit relations (Xue et al., 2015; Wang and Lan, 2015).

We followed the intuition that obtaining a significant increase in performance using traditional classifiers and feature engineering would be difficult given the effort that was previously spent on such systems. Neural-network-based classifiers present a different and less explored approach to the discourse sense problem, which can potentially lead to considerable improvement. Our system, described in this paper, takes a step in this direction.

We explore different input representation types and introduce event vectors for this task. Following the work of (Chambers and Jurafsky, 2009), we look into event chains as a way to represent structure in the discourse arguments. Then, we adapt the skip-gram approach originally used to learn word vectors from sentences (Mikolov et al., 2013b) to learn event vector representations from event sequences. To do so, we draw a clear analogy between words and events, as well as between sentences and event chains. Finally, each input relation is represented with the pre-trained event and word vectors of its arguments and a multi-layer neural network is used to classify senses.

2 System Description

The dataset used in the CoNLL shared task corresponds to the Penn Discourse Treebank (Prasad et al., 2008), in which pairs of sentences are annotated with an optional discourse connective and a sense that best explains the discourse relation between them. The annotation was done over a set of Wall Street Journal articles.

Each relation, either explicit or implicit, consists of two arguments, typically composed of short phrases and an associated sense. In the case of explicit relations, a connective is present in the

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 136–142,
Berlin, Germany, August 7-12, 2016.

text. This problem can be stated as a standard multi-class classification problem, where the inputs correspond to the argument pairs and there is a direct mapping to a finite and known set of labels.

We use two different classifiers for sense identification: a SVM classifier with linear kernel for explicit relations that uses state-of-the-art features and a multi-layer neural network for the implicit relations, which is the main focus of our submission. The following sections describe each of the systems in detail.

2.1 Explicit Discourse Relations

Explicit discourse relation detection depends on identifying explicit discourse connectives. In the sense classification task, the connective and the two corresponding arguments are supplied, therefore, we trained a linear SVM multi-class classifier to choose from 14 different senses.

We used the syntactic features described in (Lin et al., 2009; Pitler and Nenkova, 2009). We also used the connective string, PoS tags, the connective's previous word and PoS tag from Lin's features in our classifier. The features described in (Pitler and Nenkova, 2009) are extracted using constituency parse trees and consist of self-category, parent-category, left-sibling-category and right-sibling category.

(Pitler and Nenkova, 2009) has shown that using only the syntactic features, ignoring the identity of the connective gives better result. As the discourse usage of a connective may strongly rely on the syntactic context it appeared, we have added Pitler's pairwise interaction (C-Syn interaction) features between the connective C and each category feature (i.e., self-category, parent-category, left-sibling-category, right-sibling- category). The interaction features (Syn-Syn interaction) between pairs of category features are also used.

2.2 Implicit Discourse Relations

Sense classification for implicit discourse relations is notoriously hard. For this reason, we focus our efforts on this task, and explore several types of input representation and neural net architectures to deal with the challenges.

We move from the simple lexical representation of word pairs used in (Lin et al., 2009; Pitler et al., 2009), and explore the benefits of using pre-trained word vectors (Mikolov et al., 2013b) to capture combinations and similarities. Finally, we introduce the notion of an event to discourse parsing, inspired by the work of (Chambers and Jurafsky, 2009) as a way to represent structured knowledge and long range dependencies. Similar to (Modi and Titov, 2014; Pichotta and Mooney, 2016) we embed the event representation in a low dimension continuous space. More details on the definition of events and the derivation of the event vectors are given in section 2.2.1.

The sense classification task is defined over two arguments. Each argument is represented as two single vectors: a series of concatenated event vectors and a series of concatenated word vectors. A multi-layer neural network architecture receives these inputs to predict senses. The specifications of the architecture used are outlined in section 2.3.

2.2.1 Word and Event Embeddings

A word embedding is a function $W \rightarrow R^n$, mapping words to a dense low-dimensional vector space. Word embedding, recently popularized by (Mikolov et al., 2013b), can be trained to capture semantic and syntactic relationships between words, by mapping related words to vectors that lie close in the embedding vector space.

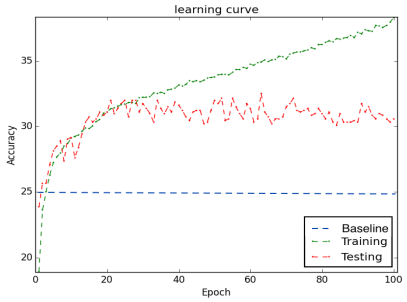
This property is often used to construct feature representations that can identify similarities and relationships between words. For example, discourse parsers often use lexical features, consisting of the product between words appearing in each of the two arguments. While such features can capture relationships between the two arguments, this representation is extremely brittle, as small variations in word usage are likely to result in lower performance. Using word embedding, instead of the arguments' words directly, can help overcome such issues.

Despite these advantages, using word embedding can potentially have several drawbacks. For example, the relationships captured between words sometimes reflect syntactic dependencies (e.g., determiners tend to be followed by nouns) rather than semantic ones, and word senses are typically ignored when word embedding are constructed. In addition, word vectors, despite their robustness, still do not capture the input argument structure.

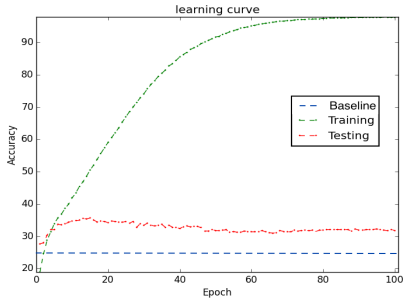
To alleviate some of these problems, we looked for a representation that can capture a higher level of abstraction of the input arguments. We propose to represent arguments as a set of events and use pre-trained event embeddings to facilitate this

	Implicit		
Class	Words	Words + Events	Number of Occurrences
Comparison.Concession	0.000	0.000	5
Comparison.Contrast	0.022	0.067	90
Contingency.Cause.Reason	0.256	0.487	78
Contingency.Cause.Result	0.036	0.143	56
Contingency.Condition	-	-	-
EntRel	0.637	0.609	215
Expansion.Alternative	-	-	-
Expansion.Alternative.Chosen alternative	0.000	0.000	2
Expansion.Conjunction	0.520	0.544	125
Expansion.Instantiation	0.000	0.163	49
Expansion.Restatement	0.173	0.260	104
Temporal.Asynchronous.Precedence	0.000	0.000	28
Temporal.Asynchronous.Succession	0.000	0.000	3
Temporal.Synchrony	0.000	0.000	20
Total	0.315	0.369	775

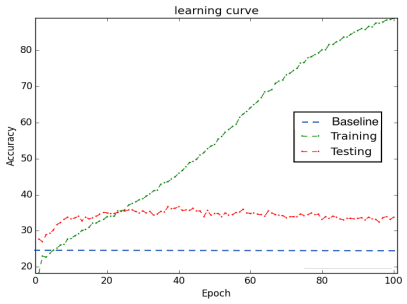
Table 1: Accuracy in the development data (Unofficial) by sense classes using different input representations: words, events and words + events



(a) Event Embeddings



(b) Word Embeddings



(c) Event+Word Embeddings

(d) Event + Word Embeddings in Adaboost learning

Figure 1: Training and testing accuracy for 100 epochs with three different implicit classifier models in Fig (a), (b), (c). Fig (d) shows the cumulative training and testing accuracy as the number of hypothesis increases from 0 to 200 in adaboost.

task. Simply put, an event can be defined as a verb and *subject* or *object* dependency relationship. An event *chain* is formed by connecting the events whose argument nodes are coreferent. We adapt the skip-gram model (Mikolov et al., 2013b) to generate event embedding by treating event chains as sentences, in which each event is a word. Note, that unlike word embedding that relies on word proximity and as a result captures syntactic information, event proximity is likely to capture temporal and causal relationships which align better with discourse relationships. Since event embedding omits much of the information contained in the input arguments, we take advantage of both word and event embeddings, and build a neural network model over both representations of the discourse arguments.

	Implicit			Explicit		
Class	Dev	Test	Blind	Dev	Test	Blind
Comparison.Concession	0.0000	0.0000	0.0000	0.5000	0.4667	0.0000
Comparison.Contrast	0.1263	0.0576	0.0000	0.9544	0.9088	0.1633
Contingency.Cause.Reason	0.3673	0.3740	0.2794	0.7568	0.8710	0.0784
Contingency.Cause.Result	0.1892	0.0896	0.0400	0.8889	0.9474	0.8571
Contingency.Condition	-	-	-	0.9318	0.8718	0.9804
EntRel	0.5647	0.5475	0.5195	-	-	-
Expansion.Alternative	-	-	-	0.9231	0.7692	0.0000
Expansion.Alternative.Chosen alternative	0.0000	0.0000	0.0000	0.9091	1.0000	-
Expansion.Conjunction	0.4069	0.3123	0.2269	0.9537	0.9495	0.6194
Expansion.Instantiation	0.2286	0.3604	0.1852	1.0000	1.0000	0.0000
Expansion.Restatement	0.2647	0.2671	0.3282	0.0000	0.4444	0.0000
Temporal.Asynchronous.Precedence	0.0000	0.3636	0.0000	0.9375	0.9459	0.0000
Temporal.Asynchronous.Succession	0.0000	0.0000	-	0.8352	0.7429	0.1562
Temporal.Synchrony	0.0000	0.0000	0.0000	0.8000	0.7742	0.4500
Total	0.3818	0.3435	0.3365	0.8968	0.8796	0.4860

Table 2: F1 score (Unofficial) by sense classes for both implicit and explicit classifier.

Pre-Training of Event Embedding The creation of event embeddings follows the Skip-gram model proposed by (Mikolov et al., 2013a). Instead of using word sequences as input to train the embeddings, we use event chains extracted by connecting events with co-referencing entities. Each entity has a chain of events and each event is represented in a form of verb and dependency pairs.

Specifically, we represent an event e as a pair $e = (v, d)$ where v denotes a verb and d denotes a grammatical dependency relation between the verb and its entity. Vector representations for events are learned from chains of events extracted from a large corpus (we used the Wikipedia dump). To start, we use Stanford CoreNLP toolkit (Manning et al., 2014) to extract dependency trees and resolve co-referent entities from the corpus. For each entity in the co-reference chain, events are extracted by looking at the adjacent verb v in the dependency tree and its correspondent grammatical dependency relation d, creating tuples (v, d) as described above. This way, chains of the form $e_i, ..., e_k$ are extracted and are used as inputs to the embedding training model.

Similar to the Word2Vec skip-gram model (Mikolov et al., 2013a), we use the following objective function.

$$J = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log \frac{exp(V'_{e_O} V_{e_I})}{\sum_{e=1}^{E} exp(V'_e V_{e_I})}$$

Where V_e is the vector representation of event e and e_I, e_O specify whether the event is an input or output (Rong, 2014). Note, that in our model, unlike the Word2Vec model that uses sentences as inputs, event chains are used as input for generating the event embedding (i.e., c refers to current event and j refers to context events in the equation above), thus capturing a higher level abstraction of the sentence semantics.

To make training feasible, we apply negative sampling following the techniques used in the word2vec model, including rare event pruning, high frequency event subsampling and a dynamic window size (Goldberg and Levy, 2014). Five negative samples are sampled for each event.

2.3 Discourse Relation Classifier

We used a Multi-layer perceptron with three hidden layers to combine the arguments' representation in our system. This layout is depicted in figure 2.

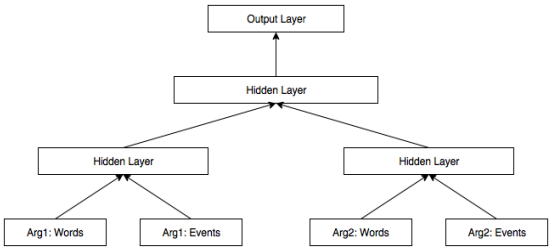


Figure 2: Three hidden layer Perceptron taking events and word vectors as input.

Two parallel hidden layers at the same level are used to combine event and word representations for every argument, this way, each hidden layer works as an abstraction of one argument. Another hidden layer is stacked on top of them to combine both arguments into a single representation. Finally an output layer with a softmax function is built on top to classify the sense. A 50% drop-out

rate is applied over all the hidden layers for the purpose of regularization. The activation function for all the hidden layers is the rectified linear function, which sets up a threshold such that all the values less than zero will be clipped to zero. In addition to this setting, we applied a drop-out rate of 20% over all the input layers. We argue that due to the high dimensionality of a combined representation of event and word embeddings, using drop-out even on the input layer can boost the model performance by avoiding overfitting.

The number of hidden units is tuned using a separate validation set. Events and word vectors are concatenated in the input layer, where the maximum number of events and words in an argument is taken from the entire date set in order to fix the size of the input and padding is performed on both sides if the number of words and/or events are less than the maximum value. In this study, we used the word embedding pre-trained on Google news corpus, which is widely used in NLP community (Mikolov et al., 2013a). For each word in this discourse parsing task, if it is in the embedding corpus, we used its mapped vector; if it is not in the embedding corpus, it is initialized to random values very close to zero. As we trained our own event embedding, we dealt with all the extracted events in a similar fashion as word embedding.

In the final model, the number of hidden nodes is 175 for agument one, 350 for agument two, and the number of units in the hidden layer stacked on them is 700.

During training, we used stochastic gradient descent with mini-batches to minimize our loss function, which we defined as the negative log-likelihood of the data. The standard back-propagation algorithm is used to compute the gradient. The whole training process is performed on an Nvidia GTX 980 GPU.

3 Experiments

Since our main focus is implicit relations, we carried out a series of experiments to test the three different input representations in the implicit sense identification task. In all these experiments, we used a neural network architecture, and used as a baseline a simple lexical classifier based on word pairs. Since during the development of the system we only had direct access to the train and development folds, most of our experiments were performed on the development data set alone.

Word pairs have been widely used for implicit sense classification (Lin et al., 2009; Pitler et al., 2009), and most systems submitted to CoNLL 2015 shared task incorporated word pairs as a fundamental part of their feature set. In table 3 we can see the aggregated results for this simple approach using Support Vector Machines on the development dataset. For this test, the top 500 word pairs ranked by information gain were used.

Input	Precision	Recall	F1
Word Pairs	0.24	0.26	0.25
Word Vectors	0.29	0.31	0.30
Event + Word Vectors	0.37	0.37	0.37

Table 3: Performance metrics on the development data for the implicit classifier

The best performing systems, however, had to go beyond simple word pairs to reach scores near 0.35. To prove the effectiveness of looking at words in a richer space, we tested a very simple neural network architecture on word vectors. This architecture incorporated only one hidden layer to combine both arguments into a single representation and an output layer with a softmax function was built on top to classify the sense. The layout for this simple architecture can be observed in figure 3. The input word vectors are concatenated in the input layer with padding and unknown words are initialized to random values very close to zero (see section 2.3).

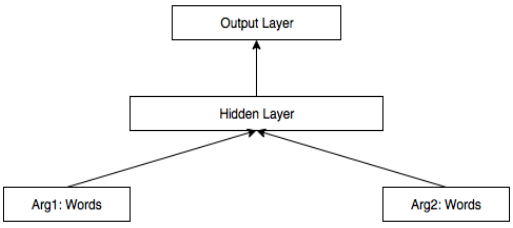


Figure 3: Single hidden layer perceptron taking word vectors as input.

Results using word vectors can be observed in table 1. We can see that there is a significant improvement in the general case, reaching to an accuracy of 0.315 in the development data. We attribute this improvement to the word vectors ability to capture similarities implicitly between words as well as providing a distributed continuous representation allowing words to be combined in the hidden layer.

Following the improvement obtained by using word vectors, we introduce event vectors into the

	Implicit			Explicit		
Class	Dev	Test	Blind	Dev	Test	Blind
Total	0.3805	0.3445	0.291	0.8968	0.8796	-

Table 4: Official TIRA F1 score for both implicit and explicit classifier.

input using the architecture described in section 2.3. After some experimentation, we decided to keep word vectors as a way to expand the information encoded in events, where the reference to the entities and other helpful lexical information is lost. Table 1 shows the improvement attained when throwing event vectors into the picture. We can observe a stable boost in performance among all classes, except EntRel, where there is a slight drop in accuracy. Total accuracy improves from 0.315 to 0.369 using event vectors, a result that is competitive with the best performing system in CoNLL 2015.

Figure 1 (a)-(c) shows the learning curve of the implicit classifier for all input types and architectures. The baseline corresponds to the word pair classifier with an accuracy of 0.25. We can observe that using word embedding overfits quickly, as the neural network starts to memorize the the training set vocabulary. Using event embedding helps combat overfitting, and the best behavior is obtained when combining the two embedding types. In this case, the learning curve in the development set reaches a higher peak. We can see the combined model overfits as the number of training epochs increases, albeit slower compared to word embedding alone. We tried to slow overfitting even further by experimenting with random sampling from training data set and combining multiple hypothesis using Adaboost, Figure 1 (d) shows the learning curve resulting from these attempts. While overfitting is indeed slower, performance suffers. We speculate that increasing the number of epochs until training accuracy reaches the optimum, may give even more competitive performance.

Tables 2 and 4 include our final results. On table 2 we can observe the performance by class on the three evaluation datasets: development, test and blind test for both the implicit and the explicit classifier. In our preliminary experiments for the implicit case, we obtained a very low score for infrequent classes. For this reason, we opted for removing infrequent classes from the training set and improved overall results, increasing F1 score from 0.36 to 0.38 for the development data.

Table 4 includes the official results obtained through TIRA (Potthast et al., 2014). Due to technical difficulties, we had to use an older model for the blind test set, that was trained over all labels (including the infrequent ones). The improvement from 0.291 to 0.3365 corresponds to the elimination of infrequent labels from the training procedure. Similarly, the system used for the blind data did not include the explicit classifier. For this reason, the result is omitted in table 4.

We looked at the class distribution in the dataset in table 1, and identified common senses that our classifier fails to distinguish. Analyzing the confusion matrix we identified the following: It is hard to differentiate Expansion.Instantiation from Expansion.Restatement and Contingency.Cause.Result from Contingency.Cause.Reason, and finally, the rest of the classes get confused with Expansion.Conjunction, which is the biggest class after EntRel.

4 Conclusion

We presented our submission for the CoNLL 2016 shared task, focusing on implict discourse sense identification[1]. We looked into deep learning approaches, as it seems that approaches that manually craft features have reached their peak. We explored different input representations for the problem and reached competitive results with CoNLL 2015 best performing system without engineering features directly.

Two types of embedding were combined: Google News pre-trained word vectors (Mikolov et al., 2013b) and our main contribution, event vectors inspired by the work of (Chambers and Jurafsky, 2009) and (Modi and Titov, 2014). We showed that event embedding for argument pairs can provide rich semantic information for the implicit discourse parsing task, significantly improving the performance of word pairs alone, even when using a very simple neural network model.

Our experiments suggest several possible future directions. First, improving event representations to include more structure seems promising. We also intend to explore using more complex learning architectures.

[1]To submit a complete system we developed a different model for explicit relations

References

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 602–610, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 343–351, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proc. of NAACL*.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 49–57, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, Phoenix, Arizona.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 13–16. Association for Computational Linguistics.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 683–691, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martin Potthast, Tim Gollub, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2014. Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling. In Evangelos Kanoulas, Mihai Lupu, Paul Clough, Mark Sanderson, Mark Hall, Allan Hanbury, and Elaine Toms, editors, *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14)*, pages 268–299, Berlin Heidelberg New York, September. Springer.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *In Proceedings of LREC*.

Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Jianxiang Wang and Man Lan. 2015. A refined end-to-end discourse parser. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 17–24, Beijing, China, July. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 1–16, Beijing, China, July. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Bonnie Webber, Attapol Rutherford, Chuan Wang, and Hongmin Wang. 2016. The conll-2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, Berlin, Germany, August. Association for Computational Linguistics.